Concurrence – Master 1
TD 3 : Sections critiques

Exercice 1 : Soit la solution suivante pour l'implémentation d'une section critique :

```
int turn = 1;
active proctype P1() {
   do :: do :: turn == 2 -> skip od;
         /* section critique */
         turn = 2;
         /* section non-critique */
   od
}
active proctype P2() {
   do :: do :: turn == 1 -> skip od;
         /* section critique */
         turn = 1;
         /* section non-critique */
   od
}
```

1. Construisez une esquise de preuve pour chaque processus, puis prouvez la non-interférence de ces preuves pour avoir la preuve globale de l'algorithme.
2. Montrez que cet algorithme satisfait les propriétés d'exclusion mutuelle et d'absence de blocage.
3. Montrez que cet algorithme ne satisfait pas les propriétés d'absence de délai et d'absence de famine.

Exercice 2 : On suppose disposer d'un ordinateur qui implémente des instructions atomiques pour incrémentation/décrementation d'une variable et qui calcule en même temps le signe du résultat. Par exemple, l'instruction de décrementation a l'effet suivant :

```
DEC(v,s): atomic { v = v - 1;
                   if :: v >= 0 -> s = 0
                      :: v < 0  -> s = 1
                   fi }
```

L'instruction INC est similaire, sauf qu'elle ajoute 1 à v.

1. En utilisant DEC et/ou INC, écrivez une solution au problème d'exclusion mutuelle entre n processus. Dans un premier temps, ne vous souciez pas de la propriété d'absence de famine. Argumenter informellement pourquoi votre solution marche et comment.
2. Construisez une esquise de preuve complète et sans interférence pour l'algorithme proposé. En utilisant cette esquise, prouvez que votre algorithme assure l'exclusion mutuelle et l'absence de blocage.

Exercice 3 : Construire une esquise de preuve (en décorant l'algorithme par des théorèmes) pour l'algorithme du boulanger (Lamport).

Correction 1 :

1.

2.

3.

Correction 2 :

1.

2.

Correction 3 :